Лебедев М.М. Взаимодействие языков программирования // Международный журнал информационных технологий и энергоэффективности. – 2019. – Т. 4 № 3(13) с. – 19-25

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.43

ВЗАИМОДЕЙСТВИЕ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Лебедев М.М.

ФГБОУ ВО «НИУ «МЭИ», Москва, Россия (111250, г. Москва, ул. Красноказарменная, д.14), e-mail: leb.dev96@gmail.com

В статье описываются возможные варианты взаимодействия нескольких языков программирования, производится их подробный разбор. Рассматриваются положительные стороны их совместного использования. Также говорится о разделении языков на компилируемые и интерпретируемые, и что понимают под промежуточными в этой классификации.

Ключевые слова: мультиязыковая разработка, взаимодействие языков, JVM, .NET CLR, COM, компиляция и интерпретация.

PROGRAMMING LANGUAGES INTERACTION

Lebedev M.M.

NRU «MPEI», Moscow, Russia (111250, Moscow, street Krasnokazarmennaya, 14), e-mail: leb.dev96@gmail.com

The paper describes in details different variants of multiple programming languages interaction. Their advantages and disadvantages are observed. What is more, in this article it is told about the subdivision of programming languages on compliable and interpretable ones and what are those which are between.

Keywords: Multilanguage programming, programming languages interaction, JVM, .NET CLR, COM, compilation and interpretation.

Для каждого языка существует своя наиболее подходящая область применения. Зачастую для разработчиков это означает, что знания одного конкретного языка программирования недостаточно, однако освоить большое их количество в совершенстве – большая проблема. Многие проекты сейчас не живут каким-то одним языком, то есть, одна из частей существует на одном языке, другая – на другом. Это очень удобно, хотя бы потому, что начинающему программисту легче определиться с выбором языка, а компаниям, участвующим в разработке найти нужных специалистов. Данная статья рассматривает, как различные языки взаимодействуют друг с другом.

Различные языки могут сосуществовать в рамках одного или нескольких процессов. Взаимодействие в рамках нескольких процессов проще: нет никакой разницы, на каком языке данные созданы, и какой их читает. Как пример можно привести пару клиент – сервер [1]. Различные процессы могут легко обмениваться данными через файл или базу данных, для этого им достаточно иметь «договорённость» о том, в каком формате и куда будут предоставляться эти данные. Также межпроцессные взаимодействия могут использовать один из следующих механизмов:

- обмен сообщениями;
- синхронизация;
- разделение памяти;
- удалённые вызовы (Remote Procedure Call).

Самым быстрым средством среди перечисленных является использование разделяемой памяти [2]. В остальных обмен происходит через ядро, что приводит к переключению контекста и снижению производительности. Для использования данного средства выделяется сегмент памяти общий для процессов, и оба процесса получают ссылку на этот участок памяти.

Удалённый вызов процедур (Remote Procedure Call) - это класс технологий, который позволяет компьютерным программам вызывать функции или процедуры в другом адресном пространстве – например, на некотором удалённом хосте. Как правило, для реализации RPC требуется общий сетевой протокол для обмена данными и оговорённый язык сериализации объектов / структур [3]. На рисунке 1 представлена схема взаимодействия процессов через удалённый вызов процедур, под «стабом» понимается фрагмент кода, который занимается конвертацией параметров перед дальнейшей передачей ядру или одному из процессов.

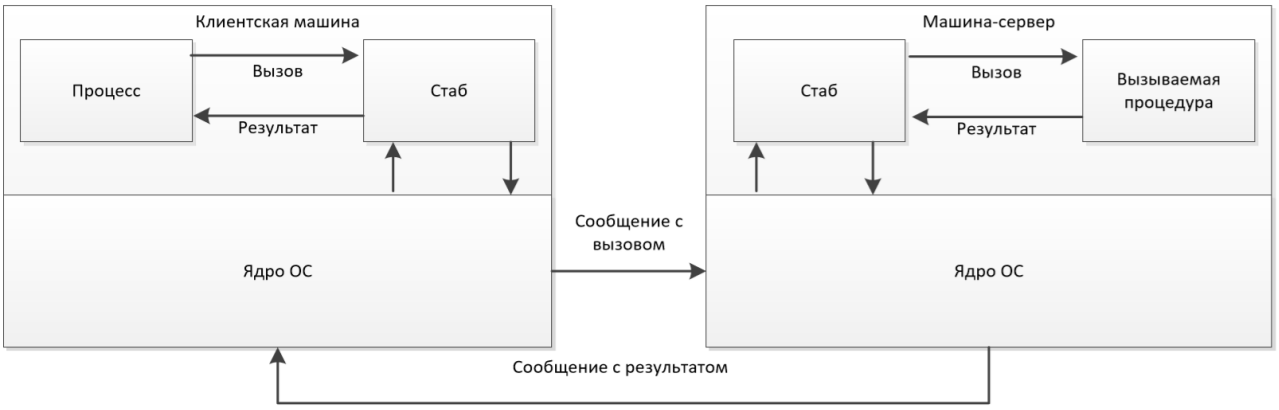

Рисунок 1 – Схема взаимодействия через удаленный вызов процедуры

Как пример обмена сообщениями, можно привести технологию обмена данными через почтовые ящики [4], данная технология доступна на компьютерах под операционной системой Windows. Почтовый ящик – это локальный псевдофайл, хранящийся в памяти. Такие ящики пропускают сообщения в одном направлении. Процесс, создавший свой почтовый ящик, является его сервером, а остальные процессы являются, в данном случае, клиентами почтового ящика. Клиенты могут посылать серверу сообщения, записывая их в почтовый ящик. Сообщение может быть послано на почтовый ящик того же компьютера, компьютера в сети или сразу на все ящики с одинаковым именем на некотором домене. Возможность посылки

широковещательных сообщений – одна из главных особенностей именно технологии почтовых ящиков. Пришедшие сообщения дописываются и хранятся до «прочтения» их сервером. Для обеспечения двунаправленности двум процессам необходимо завести два собственных ящика. Другим более известным примером являются каналы – простое средство для взаимодействия процессов систем на базе ОС UNIX. Различают именованные и неименованные каналы. Неименованные представляют собой сущность, в которую можно помещать и извлекать данные через два файловых дескриптора (для чтения и записи соответственно) [5]. Именованные каналы представляют собой файлы отдельного типа, доступ к которым любой из процессов может получить в любое время, используя имя этого канала [6].

Когда речь идёт о взаимодействии в рамках одного процесса - функциям и подпрограммам нужно уметь вызывать друг друга, для чего и вводят стандарты вызова. Для языков семейства C и Pascal обычно применяется стандарт бинарных соглашений. Другим примером стандартов являются COM-объекты, которые могут быть написаны на многих языках.

Component Object Model — это технологический стандарт от компании Microsoft, закрепившийся в основном на ОС семейства Windows. Он предназначен для создания ПО на основе взаимодействующих компонентов объекта. Каждый из объектов может быть одновременно использован несколькими процессами [7]. Программы, построенные на стандарте, фактически представляют собой набор взаимодействующих между собой COM-компонентов. Взаимодействие происходит через COM-интерфейсы, по умолчанию каждый компонент должен, как минимум, переопределять интерфейс «IUnknown».

Как уже говорилось выше, процессы могут вызывать функции на других языках, обмениваясь результатами их выполнения. Соглашение о вызове - это описание технических особенностей вызова подпрограмм, определяющее:

- способы передачи параметров подпрограммам (их порядок помещения в стек);
- способы вызова (передачи управления);
- способы передачи результатов обратно в точку вызова;
- способы возврата (передачи управления).

Рассмотрим особенности некоторых из применяемых соглашений [8]:

- Cdecl – стандарт языков C. Передача аргументов - справа налево. Вызывающая программа должна сама очищать стек, сохранять, а затем восстанавливать значения регистров.
- Pascal – используется компиляторами языка Паскаль. Аргументы передаются через стек, слева направо. Результат возвращается через параметр Result.
- Stdcall или Winapi — применяется в ОС Windows для вызова функций WinAPI. Аргументы передаются через стек, справа налево. Вызываемая подпрограмма должна очищать стек.
- Fastcall — самый быстрый способ, так как аргументы передаются через регистры (если их недостаточно, то используют стек). Вызов не стандартизирован.
- Safecall — соглашение, используемое для вызова методов интерфейсов COM. Методы представляют собой функции, возвращающие результат типа HRESULT, который затем должен быть проанализирован.
- Thiscall – используется компиляторами C++ при вызове методов классов. Почти не отличается от Cdecl.

Стоит отдельно отметить ещё один способ взаимодействия языков в рамках нескольких процессов – когда языки компилируются в общий промежуточный код. Например, языки .NET и языки для виртуальной машины Java «собираются» в единый исходный код. Поэтому объекты, созданные на Java доступны, например, на Scala – доступность определяется на уровне метаданных виртуальной машины. Чтобы подробнее рассмотреть данный тип взаимодействия необходимо вспомнить об условном делении языков на компилируемые и интерпретируемые. [9].

Программы на компилируемых языках собираются программой-компилятором в исполняемый модуль, содержащий набор инструкций для некоторого типа процессора. Для выполнения кода на интерпретируемом языке требуется программа-посредник (интерпретатор): она выполняет исходный код программы без перевода. На рисунке 2 представлена схема, отображающая принципиальное отличие компиляции от интерпретации.

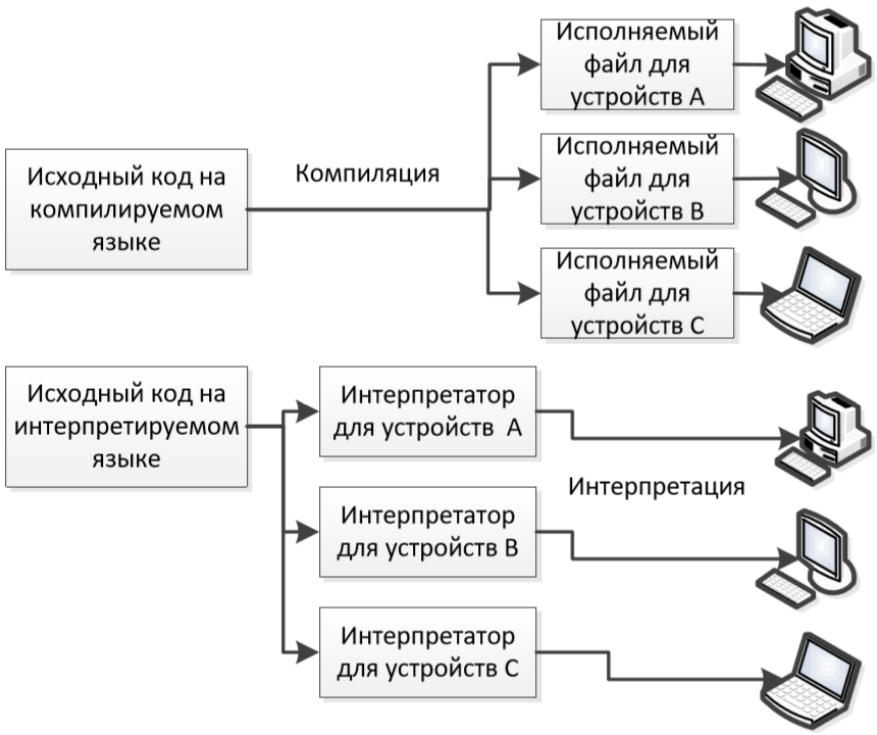

Рисунок 2 – Компиляция и интерпретация программ

JVM (Java Virtual Machine) – или виртуальная машина Java, исполняет собственный байт код, который собирается Java компиляторами. В Java код могут быть собраны программы на Scala, Ada и т. д. JVM можно установить практически на любое устройство. То есть программа, созданная и скомпилированная на одном устройстве на языке семейства Java, может быть выполнена практически где угодно. [10]

Подобно JVM, .NET CLR представляет собой виртуальную машину, интерпретирующую байтовый код. Эти байтовые коды отличаются, однако машины стремятся предоставлять схожие функции. То, что у JVM называется «Java байт-кодом» у

.NET CLR называют инструкциями промежуточного языка «Intermediate Language» (IL). [11] Обе виртуальные машины имеют возможность компилировать свой входной байт-код на машинный язык компьютера, на котором они запущены прямо во время выполнения – это

называется «Just In Time Compilation» (JIT) [12], а полученный выходной код называют JIT-кодом. Таким образом, языки семейств Java и .NET могут взаимодействовать на уровне своей виртуальной машины.

Скомпоновав собранную информацию, получим диаграмму классификации взаимодействий языков программирования. Данная диаграмма представлена на рисунке 3.

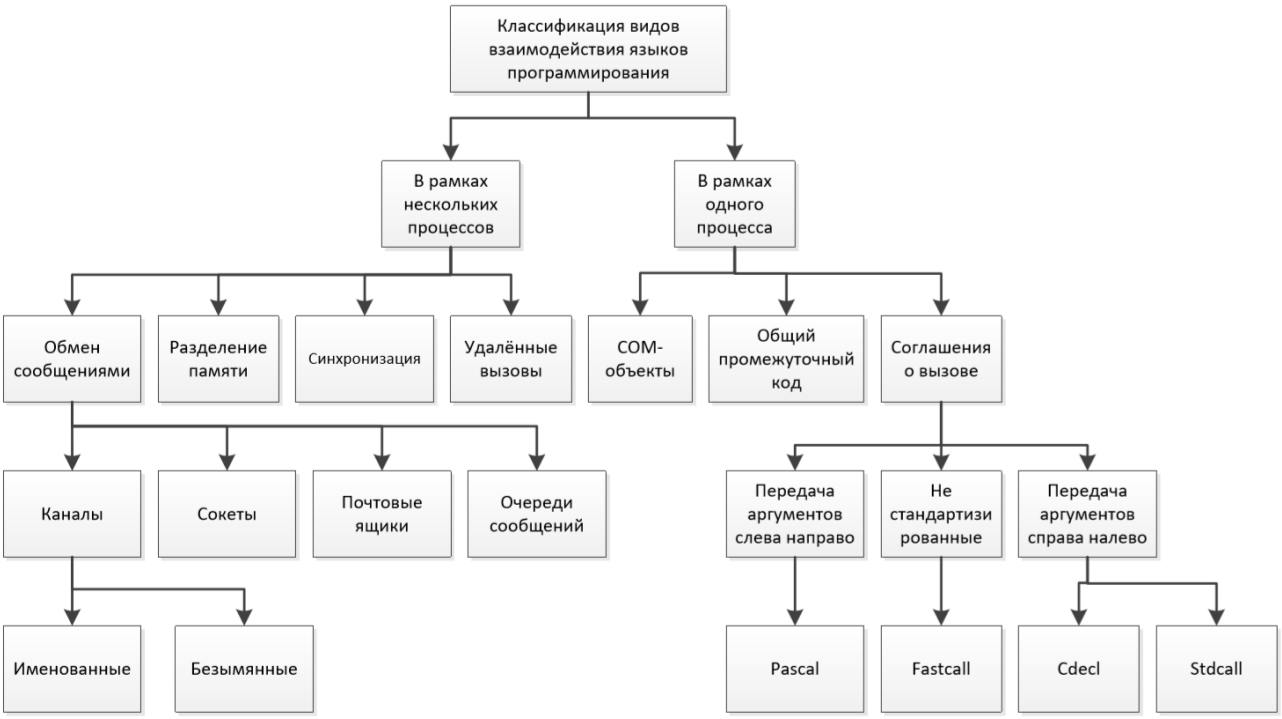

Рисунок 3 – Классификация взаимодействия языков программирования

Таким образом, вид взаимодействия в первую очередь определяется тем, в рамках скольких процессов происходит взаимодействие. Далее классификация определяет метод взаимодействия по способу передачи данных (и / или управления). Некоторые из способов, в свою очередь, также имеют подвиды, и их можно также классифицировать по характерным признакам.

Список литературы

1. Анатольев А. Г., Конспект лекций по курсу «Сетевые технологии». Омск: ОмГТУ, 2013. 79 с.
2. Иртегов Д. В., Конспект лекций по курсу «Системы семейства UNIX». Новосибирск: НГУ, 2015. 146 с.
3. Межпроцессное взаимодействие (Операционные системы) [Электронный ресурс] / Национальная библиотека им. Н. Э. Баумана – Режим доступа: https://ru.bmstu.wiki/Межпроцессное_взаимодействие_(Операционные_Системы), свободный. (Дата обращения: 25.07.2019 г.).
4. Джонсон М. Харт, Системное программирование в среде Windows: пер. с англ. - М.: Издательский дом "Вильямс", 2005. - 592 с.

5. Именованный канал [Электронный ресурс] / Википедия – свободная энциклопедия – Режим доступа: https://ru.wikipedia.org/wiki/Именованный_канал, свободный. (Дата обращения: 25.07.2019 г.).
6. Неименованный канал [Электронный ресурс] / Википедия – свободная энциклопедия – Режим доступа: https://ru.wikipedia.org/wiki/Неименованный_канал, свободный. (Дата обращения: 25.07.2019 г.).
7. Component Object Model [Электронный ресурс] / Википедия – свободная энциклопедия – Режим доступа: https://ru.wikipedia.org/wiki/Component_Object_Model, свободный. (Дата обращения: 25.07.2019 г.).
8. Соглашение о вызове [Электронный ресурс] / Википедия – свободная энциклопедия – Режим доступа: https://ru.wikipedia.org/wiki/Соглашение_о_вызове, свободный. (Дата обращения: 25.07.2019 г.).
9. Суховилов Б. М., Конспект лекций по курсу «Информатика». Челябинск: ЮУрГУ, 2015. 59 с.
10. Что такое JVM? Знакомство с виртуальной машиной Java [Электронный ресурс] / Topjava.ru - курсы программирования на java – Режим доступа: https://topjava.ru/blog/what-is-the-jvm, свободный. (Дата обращения: 25.07.2019 г.).
11. Обзор среды CLR — .NET Framework | Microsoft Docs [Электронный ресурс] / Microsoft – официальная страница – Режим доступа: https://docs.microsoft.com/ruru/dotnet/standard/clr, свободный. (Дата обращения: 25.07.2019 г).
12. JIT для начинающих – devSchacht – Medium [Электронный ресурс] / Medium – a place to read and write big ideas and important stories – Режим доступа: https://medium.com/devschacht/how-to-start-jitting-ee9fcbc9065a, свободный. (Дата обращения: 25.07.2019 г).

References

1. Anatolyev A. G., Lecture notes on «Net technologies». Omsk: OmSTU, 2013. 79 p.
2. Irtegov D. V., Lecture notes on «UNIX family systems». Novosibirsk: NSU, 2015. 146 p.
3. Inter-process communication (operational systems) [Electronic resource] / National library of N. E. Bauman – Access mode: https://ru.bmstu.wiki/Межпроцессное_взаимодействие_(Операционные_Системы), free. (Request Date: 25.07.2019).
4. Johnson M. Hart, Windows system programming. 3-rd ed. The Addison-Wesley Microsoft Technology Series, 2005. 656 p.
5. Named pipe [Electronic resource] / Wikipedia, the free encyclopedia – Access mode: https://en.wikipedia.org/wiki/Named_pipe, free. (Request Date: 25.07.2019).
6. Anonymus pipe [Electronic resource] / Wikipedia, the free encyclopedia – Access mode: https://en.wikipedia.org/wiki/Anonymous_pipe, free. (Request Date: 25.07.2019).
7. Component Object Model [Electronic resource] / Wikipedia, the free encyclopedia – Access mode: https://ru.wikipedia.org/wiki/Component_Object_Model, free. (Request Date: 25.07.2019).
8. Calling convention [Electronic resource] / Wikipedia, the free encyclopedia – Access mode: https://en.wikipedia.org/wiki/Calling_convention, free. (Request Date: 25.07.2019).
9. Suhilov B. M., Lecture notes on «Informatics». Chelyabinsk: SUSU, 2015. 59 p.
10. What is JVM? Getting acquainted to Java [Electronic resource] / Topjava.ru – java programming courses – Access mode: https://topjava.ru/blog/what-is-the-jvm, free. (Request Date: 25.07.2019).

11. CLR overview — .NET Framework | Microsoft Docs [Electronic resource] / Microsoft – official page – Access mode: https://docs.microsoft.com/ru-ru/dotnet/standard/clr, free. (Request Date: 25.07.2019).
12. JIT for beginners – devSchacht – Medium [Electronic resource] / Medium – a place to read and write big ideas and important stories – Access mode: https://medium.com/devschacht/howto-start-jitting-ee9fcbc9065a, free. (Request Date: 25.07.2019)